Fondamenti Teorici

Antonio Pescapè e Marcello Esposito

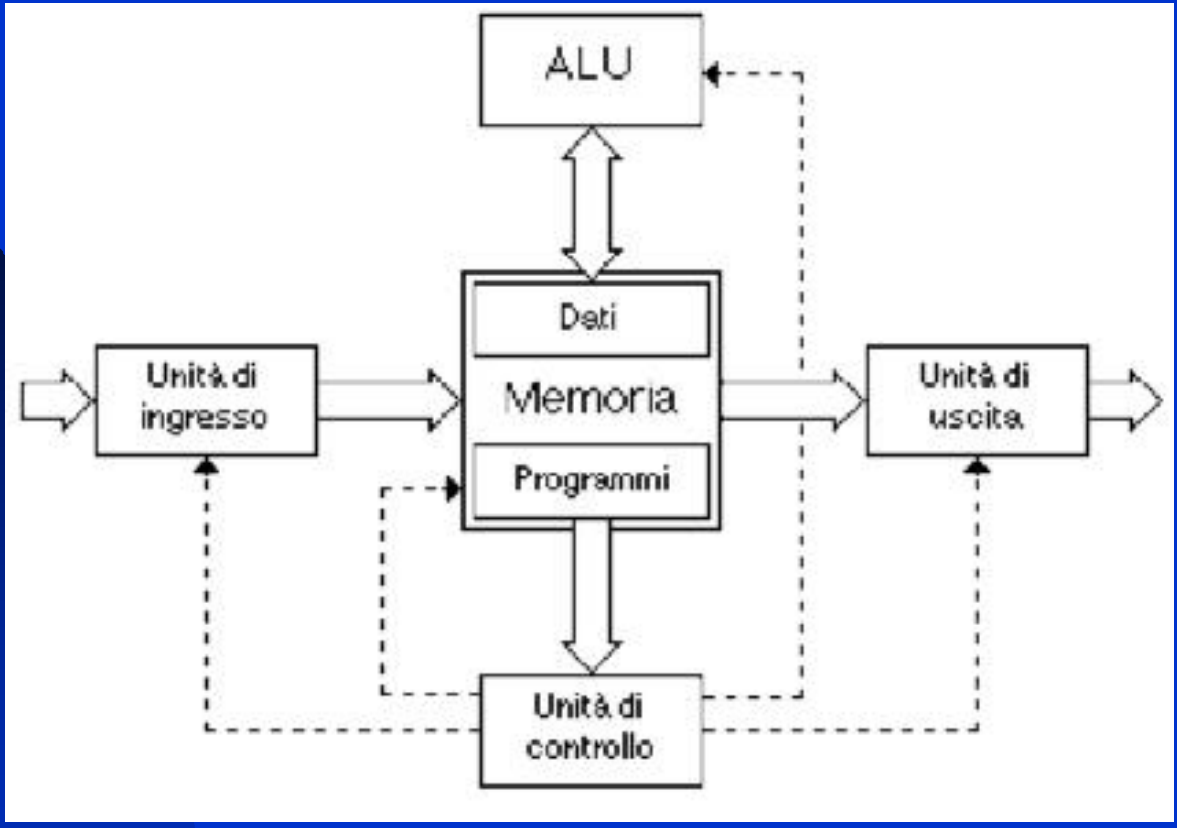
Parte Seconda

v2.0

Agenda

- Modello di Von Neumann
- Algoritmo del Processore
- Linguaggio Macchina e Linguaggio Assembler
- Hardware e Software
- Compilatori e Interpreti
- Organi di un calcolatore
- Architettura di un processore

Modello di Von Neumann (1)



Il modello di Von Neumann è il più prossimo all'effettiva struttura di un calcolatore ed è stato utilizzato come modello costruttivo dei primi elaboratori

Modello di Von Neumann (2)

- **L'unità di ingresso permette l'immissione dei dati nella memoria e anche del programma in una fase detta di "caricamento"**

- **L'unità di memoria memorizza istruzioni e dati (iniziali, intermedi e finali)**

- **L'unità di controllo presiede a tutte le operazioni del calcolatore; essa preleva, una alla volta, le istruzioni da eseguire dalla memoria e invia agli altri organi i segnali per l'esecuzione delle singole operazioni**

Modello di Von Neumann (3)

- **L'unità logico-aritmetica è in grado di eseguire su richiesta della control unit le operazioni aritmetiche e logiche (operazioni aritmetiche (somme, sottrazioni, ...), operazioni logiche (and, or, not, ...), operazioni di spostamento di bit (shift, rotate, ...))**
- **L'unità di uscita per la rappresentazione dei risultati**
- **Le linee di collegamento prendono il nome di Bus di comunicazione: informazioni (dati e/ o istruzioni) e segnali di controllo**
- **L'unità di Controllo e la ALU costituiscono la CPU (Central Processing Unit) o Processore**

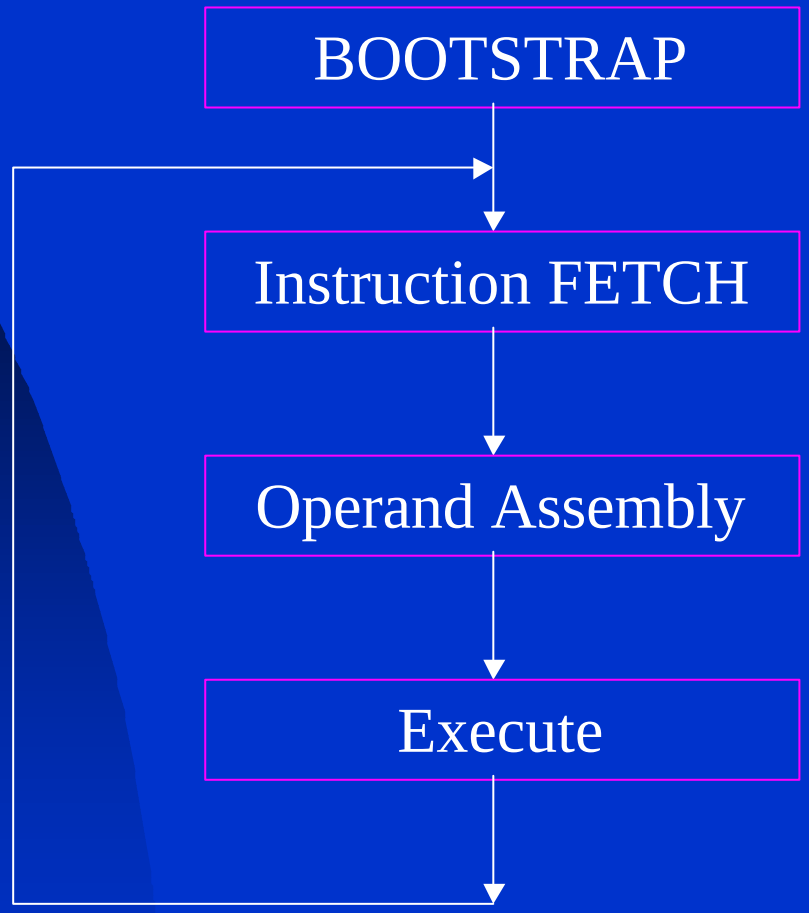
Modello di Von Neumann (4)

- **Un programma per essere eseguito deve trovarsi in memoria**

- **Un programma viene prelevato dall'unità di ingresso e caricato in memoria. Dopodiché, ad opera dell'unità di controllo che interpreta le successive istruzioni ed attiva opportunamente le altre unità, il programma viene eseguito.**

- **Mentre il programma si trova in memoria per essere eseguito, l'unità di controllo opera alternando in ciclo fasi dette del "ciclo del processore": fetch, operand assembly ed execute.**

Algoritmo del Processore (1)



Algoritmo del Processore (2)

- **L'unità di controllo accede ad una istruzione (determinata dall'istruzione precedentemente eseguita), prelevandola dalla memoria e trasportandola nei propri registri**

- **In funzione del tipo di istruzione prelevata si procede alla preparazione degli operandi**

- **L'istruzione individuata viene interpretata ed eseguita con l'ausilio a seconda dei casi delle altre unità**

Al termine dell'esecuzione, l'unità di controllo riprende, ciclicamente, un nuovo accesso e una nuova esecuzione

Algoritmo del Processore (3)

A questo punto possiamo affermare che un elaboratore è un automa, in quanto esso esegue un algoritmo (l'algoritmo del processore):

- **in un numero finito di passi**
- **in sequenza (ciclicamente)**
- **automaticamente**
- **deterministicamente**

Istruzioni Assembler

- Le istruzioni prelevate nella fase di fetch sono istruzioni in linguaggio assembler:
  - MOVE A,B: muovi i dati dalla locazione di memoria all'indirizzo A alla locazione di memoria all'indirizzo B
  - ADD X,Y: somma il contenuto della memoria all'indirizzo X al contenuto della memoria all'indirizzo Y e metti il risultato in Y
  - JUMP Z: salta alla prossima istruzione alla locazione di memoria Z

Istruzioni Macchina

Stringhe di 0 e di 1 che hanno un significato particolare per l'unità di controllo dell'elaboratore su cui vengono eseguite

Ogni CPU:

- ha il proprio insieme di istruzioni macchina
- è in grado di eseguire solo le istruzioni appartenenti a tale insieme

Le istruzioni macchina dipendono dalla realizzazione fisica della CPU: il loro numero e il loro formato variano al variare dell'elaboratore (o della famiglia di elaboratori)

Linguaggio Macchina

Per scrivere programmi in **linguaggio macchina** è necessario:

- conoscere l'architettura interna del processore che si utilizza
- conoscere i dettagli relativi alla scrittura delle singole istruzioni:
  - codici numerici delle istruzioni macchina
  - formato interno delle istruzioni macchina
  - rappresentazione degli operandi, ...
- gestire direttamente gli indirizzi in memoria per
  - riferimenti ai dati
  - salti al codice

Linguaggio Assembler

Un Programma scritto in linguaggio assembler è una sequenza di istruzioni elementari in corrispondenza 1: 1 con istruzioni macchina

Vantaggio :

utilizzo di **nomi simbolici** al posto di

- codici numerici
- indirizzi di memoria (nomi di variabili e label)

Linguaggio Assembler

CLASSIFICAZIONE DELLE ISTRUZIONI

- **istruzioni per il trasferimento dati registro/ memoria - registro/ memoria:**
  - LOAD (da memoria principale a registro)
  - STORE (da registro a memoria principale)
  - MOVE (trasferimento generico)
- **istruzioni aritmetiche, operazioni aritmetiche su interi (e reali):**
  - somma (ADD) e sottrazione (SUB)
  - incremento (INC) e decremento (DEC) di 1
  - moltiplicazione (MUL) e divisione (DIV)
- **istruzioni per la manipolazione di bit :**
  - operazioni logiche (AND, OR, XOR e NOT)
  - operazioni di scorrimento (SHIFT) e rotazione (ROTATE) di bit

Linguaggio Assembler

CLASSIFICAZIONE DELLE ISTRUZIONI

- **istruzioni per il trasferimento del controllo :**
  - **- salti incondizionati e condizionati (JUMP, BRANCH)**
  - **- chiamate a sottoprogramma (CALL)**
  - **- ritorno da sottoprogramma (RET)**
  - **- uniche istruzioni che modificano PC**
  - **- strutture di controllo dei linguaggi ad alto livello -> salti incondizionati e condizionati**
- **istruzioni per il controllo del processore :**
  - **- HALT**
- **istruzioni per l'input/ output :**
  - **-IN e OUT**

Linguaggio Assembler

FORMATO DELLE ISTRUZIONI

- istruzione divisa in varie sezioni
- ogni sezione ha un particolare significato per l'unità di controllo

Prima sezione : codice operativo dell'istruzione

codice numerico che indica all'unità di controllo quale particolare istruzione deve essere eseguita

Resto dell'istruzione : informazioni necessarie per determinare la collocazione degli operandi dell'istruzione

Il termine operando indica indistintamente:

- un **operando sorgente** da leggere da un registro o dalla memoria
- un **operando destinazione** da memorizzare in un registro o in memoria
- l' **indirizzo dell'istruzione** a cui trasferire il controllo, nel caso di istruzioni di salto

Linguaggio Assembler

Classificazione in base al numero di operandi espliciti :

- **istruzioni a tre operandi** : ad esempio, un'istruzione di somma
- **istruzioni a due operandi :** ad esempio, un'istruzione di trasferimento
- **istruzioni ad un operando :** ad esempio, un'istruzione di salto
- **istruzioni senza operandi :** ad esempio, un'istruzione HALT

Utilizzo di **operandi impliciti:**

Un'istruzione di somma può utilizzare il registro accumulatore come **sorgente implicita** di uno degli addendi e **destinazione implicita** del risultato

Un'istruzione di trasferimento dalla memoria a un registro macchina può utilizzare il registro accumulatore come **destinazione implicita**

Hardware e Software (1)

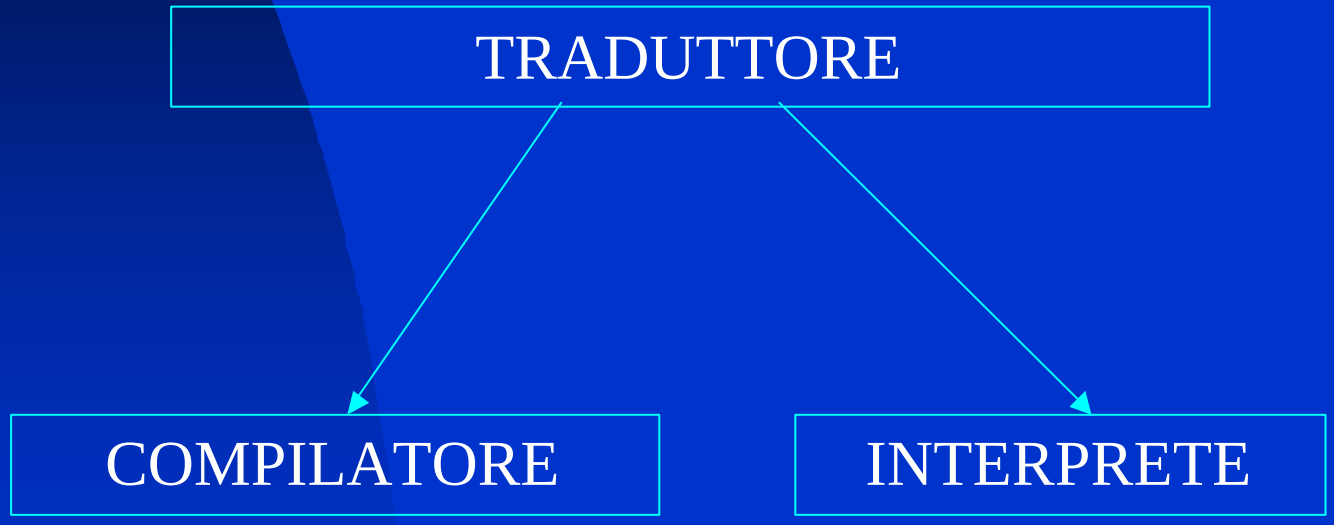
- **Un sistema di elaborazione è classicamente suddiviso in due macro-strati :**
  - **HARDWARE : insieme dei circuiti, componenti elettrici ed elettronici, componenti meccanici**
  - **SOFTWARE : insieme di programmi operanti sulla macchina**
- **Lo strato hardware è costituito dalla macchina di Von Neumann. Tale strato è anche detto macchina base.**
- **Il linguaggio della macchina base è il linguaggio macchina**

Hardware e Software (2)

- Un linguaggio macchina, pur essendo di basso livello, permette di risolvere problemi di grossa complessità. E' pur vero però che lo sforzo di programmazione può essere proibitivo.
- I sistemi di elaborazione vengono pertanto completati con programmi del software di base che ne consentono un uso più semplice e rapido
- Aggiungendo il software di base il calcolatore appare all'utente come una macchina "diversa" che prende il nome di macchina virtuale.
- Il software di base si serve di appositi programmi traduttori per simulare le diverse macchine virtuali sull'unica macchina base.

Compilatori e Interpreti (1)

I traduttori di linguaggio operano secondo due tecniche distinte: compilazione e interpretazione



Compilatori e Interpreti (1)

Con la compilazione (operata da un compilatore) il programma in linguaggio simbolico viene prima tradotto in linguaggio macchina e successivamente il programma in linguaggio macchina viene eseguito dalla macchina base. L'elaborazione pertanto si compie in tempi distinti :

Compilatori e Interpreti (2)

- **Con l'interpretazione (eseguita da un interprete) non si distinguono il tempo di esecuzione del traduttore da quello del programma specifico**

- **L'esecuzione dell'interprete consiste nell'interpretare, istruzione per istruzione, il programma in esame, richiedendo alla macchina base di eseguire, per ciascuna di esse, le azioni elaborative equivalenti.**

- **Il programma da interpretare resta nella sua forma originaria anche durante la sua esecuzione**

Compilatori e Interpreti (3)

Per fissare le idee facciamo un esempio:

COMPILATORE

Dobbiamo sottoporre un nostro curriculum, in inglese, ad una azienda, ma non conosciamo l'inglese. Abbiamo bisogno di un traduttore che traduca quanto scritto da noi dall'italiano all'inglese. Faremo i seguenti passi:

- **contattiamo il traduttore**
- **il traduttore riceve il testo da tradurre**
- **il traduttore fornisce il testo tradotto**
- **possiamo sottoporre il nostro curriculum all'azienda**

L'analogia con il diagramma temporale precedente è evidente.

Compilatori e Interpreti (4)

INTERPRETE

Dobbiamo incontrare un manager cinese per motivi di lavoro ma non conosciamo il cinese. Abbiamo bisogno di un interprete che traduca il nostro dialogo. Faremo i seguenti passi:

- **contattiamo l'interprete**
- **noi parliamo in italiano, in presenza dell'interprete**
- ***contemporaneamente* l'interprete comunica al manager cinese quanto detto da noi e viceversa**

Il compito dell'interprete si svolge contestualmente all'incontro col manager cinese.

Compilatori e Interpreti (5)

Confronto Interprete/Compilatore

- Un programma, una volta compilato, può essere eseguito un numero arbitrario di volte senza la necessità di ulteriori traduzioni.

- Un programma scritto in un linguaggio interpretato, deve essere tradotto nuovamente all'atto di ciascuna esecuzione.

- Detto t_t il tempo di traduzione e t_e il tempo di esecuzione, per eseguire un programma n volte, il tempo necessario sarà:
  - $t_t + n \bullet t_e$, nel caso di programma compilato
  - $n \bullet t_t + n \bullet t_e$, nel caso di programma interpretato

  E questo nel caso migliore. Nel caso in cui ci siano dei cicli nel programma, questi vengono interpretati ad ogni occorrenza del ciclo.

Compilatori e Interpreti (6)

Confronto Interprete/Compilatore

- L'esecuzione di un programma compilato risulta più veloce rispetto a quella di un programma interpretato.

- I linguaggi interpretati sono tipicamente più flessibili e semplici da utilizzare (nei linguaggi compilati esistono limitazioni alla semantica dei costrutti: presenza in memoria dei dati, dimensioni statiche degli array, ecc.

- Per distribuire un programma interpretato si deve necessariamente distribuire il codice sorgente, rendendo possibili operazioni di plagio

- Rilevazione degli errori al tempo di traduzione

Sistema a Programma Registrabile

- **Un elaboratore numerico opera sulle rappresentazioni dei valori delle informazioni e le trasforma in rappresentazioni dei risultati. Queste informazioni sono contenute nei registri.**

- **La memoria dell'elaboratore è costituita da un numero finito di registri numerabili e singolarmente indirizzabili.**

- **Il programma è una sequenza di istruzioni che definisce una elaborazione che il calcolatore deve eseguire. Esso può essere memorizzato e richiamato ogni volta che si voglia eseguire l'elaborazione ad esso associata.**

Sistema a Programma Registrabile

- **Un elaboratore è in grado di memorizzare più programmi**

- **La memorizzazione dei programmi è la caratteristica fondamentale di un elaboratore: sostituendo il programma in memoria si definisce l'elaborazione da eseguire**

Macchina polifunzionale

Organi di un calcolatore (1)

Unità di Ingresso

- tastiera
- dischi, dischetti (floppy) e nastri magnetici
- schede magnetiche (badge)
- penna fotoelettrica
- convertitori A/D
- scanner e lettori ottici
- apparati di trasmissione e ricezione dati (modem)
- unità vocali

Organi di un calcolatore (2)

Unità di Uscita

- monitor
- stampanti
- plotter
- unità audio
- convertitori D/A
- dischi, dischetti (floppy) e nastri magnetici
- apparati di trasmissione e ricezione dati (modem)

Organi di un calcolatore (3)

La memoria centrale

- costituita da un insieme ordinato di registri (celle o locazioni)
- alle locazioni è associato un indirizzo univoco (indirizzo di memoria)
- in ciascuna cella è memorizzato un dato o un'istruzione
- l'operazione di lettura (prelievo) non è distruttiva
- l'operazione di scrittura (registrazione) è distruttiva
- tempo di accesso e capacità

Organi di un calcolatore (4)

La memoria centrale, tipologie di memoria

- Random Access Memory (RAM)
- Read Only Memory (ROM)
- Programmable ROM (PROM)
- Erasable Programmable ROM (EPROM)
- Electtrically Erasable Programmable ROM (EEPROM)

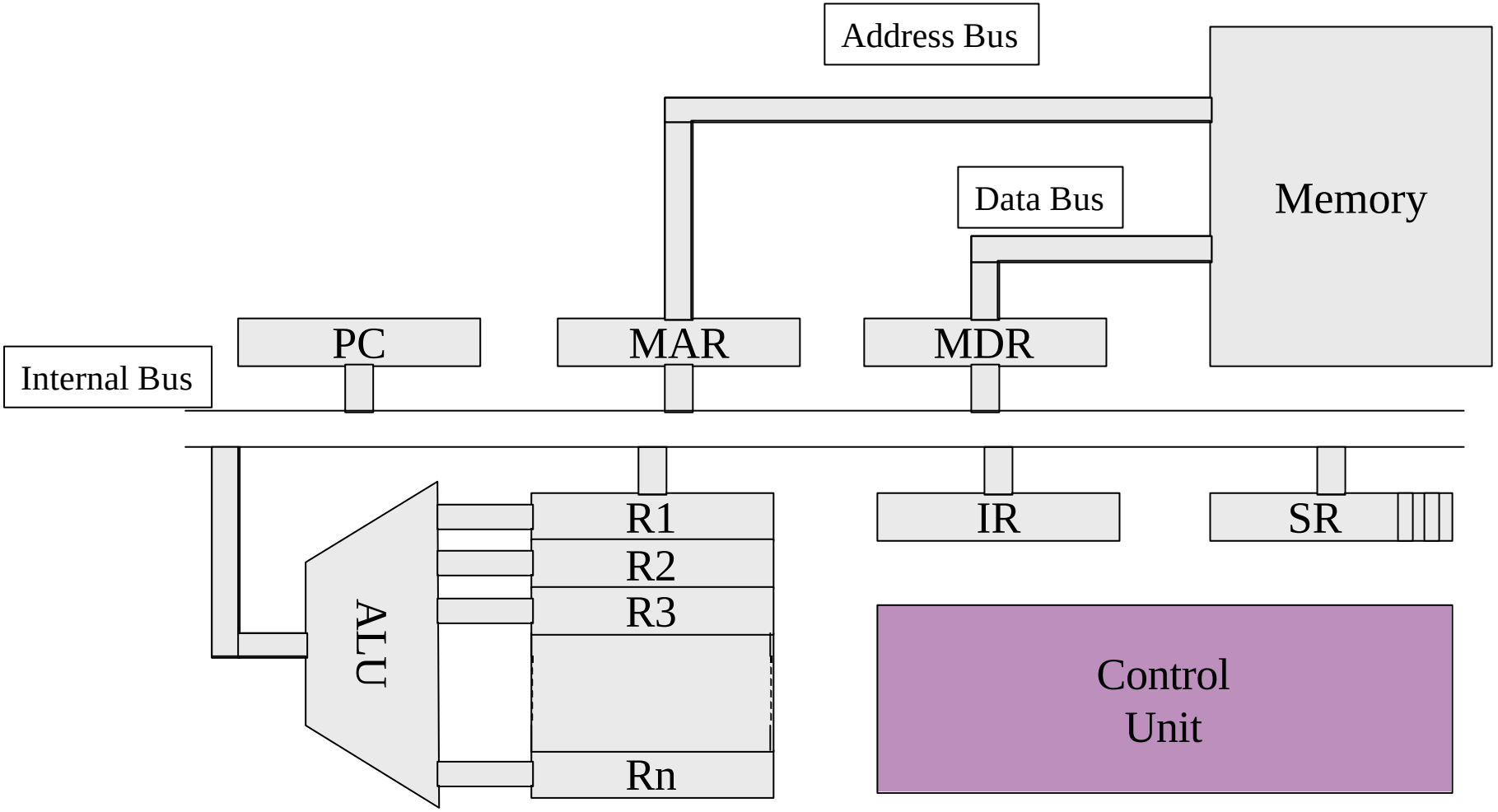
Organi di un calcolatore (5)

La memoria di massa

- Dischi magnetici fissi (hard disk)
- Dischi magnetici rimovibili (floppy disk)
- Dischi ottici rimovibili a sola lettura (CD-ROM)
- Dischi ottici rimovibili a lettura e scrittura (CD-RW)
- Nastri magnetici

Architettura di un processore

Architettura di un processore basato su registri generali



I componenti

- L'architettura di un processore basata su registri generali è costituita da una serie di componenti fondamentali:
  - l'Arithmetic Logic Unit (ALU);
  - la Control Unit (CU);
  - la memoria;
  - l'Internal Bus, il Data Bus e l'Address Bus;
  - i registri del processore (o interni):
    - il Program Counter (PC)
    - il Memory Address Register (MAR)
    - il Memory Data Register (MDR)
    - l'Instruction Register (IR)
    - lo Status Register (SR)
    - i cosiddetti registri generali R1.. Rn.
- I registri sono organi di memoria atti a memorizzare una serie di bit. Valori tipici del numero di bit che essi memorizzano sono: 8, 16, 32 o 64.

L'Internal Bus

- È un canale di comunicazione condiviso dai vari componenti ed attraverso cui essi possono dialogare. In questo contesto, il dialogo consiste nello scambio di dati binari tra registri secondo una modalità parallela. Ciò significa che un certo numero di bit viene contemporaneamente trasferito attraverso il bus da un registro mittente ad un registro destinatario.
- Durante un'operazione di trasferimento, i due registri implicati nella comunicazione si trovano in uno stato di lettura (destinatario) e scrittura (mittente) in modo tale da poter acquisire il dato presente sul bus e da potercelo scrivere, rispettivamente. Tutti gli altri registri sono in uno stato di "riposo" nel quale non possono né leggere i dati che circolano sul bus né influenzare lo stato del bus con i dati che contengono.
- Il numero di bit contemporaneamente trasferiti, indica il parallelismo del bus ed è pari al numero di bit contenuti in un singolo registro. Esso caratterizza anche il parallelismo interno del processore.

Il Data Bus e il registro MDR

- Il Data Bus è un bus che collega la memoria con il registro MDR. Serve a trasferire dati in entrambi i sensi, sempre secondo una modalità parallela.

- Tutti i dati e le istruzioni che dalla memoria devono essere elaborati nel processore, transitano per il registro MDR e successivamente, da questo, raggiungono gli opportuni registri per l'elaborazione vera e propria.

- Analogamente, tutti i risultati di un'elaborazione che devono essere immagazzinati in memoria, transitano prima per il registro MDR e da esso raggiungono poi la esatta locazione di memoria.

L’Address Bus e il registro MAR

- Durante un accesso alla memoria, sia in fase di lettura che in fase di scrittura, il registro MAR contiene l’indirizzo della locazione di memoria che viene acceduta.

- Questo indirizzo, trasferito all’organo memoria attraverso l’Address Bus, abilita alla comunicazione una sola tra tutte le locazioni disponibili (tipicamente in numero molto elevato).

Il registro PC

- Il valore memorizzato nel registro PC rappresenta per definizione l'indirizzo della locazione di memoria contenente la successiva istruzione da eseguire.
- Esso viene interrogato tipicamente all'inizio di ogni fase di fetch ed immediatamente dopo viene aggiornato alla locazione di memoria "seguente"; in questo modo lo si prepara per il prelievo della successiva istruzione.
- Può accadere comunque che l'istruzione prelevata rientri nella categoria delle istruzioni di salto: in questo caso si procede ad un ulteriore aggiornamento del PC durante la fase di execute dell'istruzione.
- Da questo deriva che lo scopo di un'istruzione di salto (condizionato) è esclusivamente quello di alterare (eventualmente) il valore del PC. Spesso il registro PC è chiamato anche IP (Instruction Pointer).

Il registro IR

- Questo registro ha il compito di accogliere dalla memoria (attraverso il MDR), durante una fase di fetch, l'istruzione da eseguire, quella cioè puntata dal PC.
- Una volta in questo registro, l'istruzione deve essere interpretata dall'unità di controllo per procedere alla eventuale fase di preparazione degli operandi ed alla fase di esecuzione.

La ALU

- L’unità logico-aritmetica è l’organo deputato allo svolgimento di operazioni aritmetiche e di confronti logici. Essa preleva gli operandi tipicamente dai registri generali, così come nei registri generali depone i risultati dei calcoli. In seguito ad un calcolo ha anche il compito di impostare alcuni dei flags del SR in modo da tenere traccia di determinati eventi.

Il registro SR

- Lo Status Register è un registro che memorizza una serie di bit indicativi dello stato corrente del processore.
- Può indicare, ad esempio, se il risultato dell'ultima operazione aritmetica effettuata dall'ALU ha dato risultato nullo, o se ha generato un riporto.

I registri generali

- I registri generali non hanno un preciso ruolo come gli altri, e da ciò scaturisce il loro nome.
- Sono utilizzati per contenere i dati in transito per un'elaborazione, come per es.:
  - gli addendi di un'addizione che l'ALU sta per effettuare;
  - il risultato di un calcolo che l'ALU ha effettuato;
  - un indirizzo di memoria in cui si trova un dato che dovrà essere acceduto in seguito.
- Un numero elevato di tali registri conferisce maggiore flessibilità nella programmazione, ma complica la struttura del processore dal punto di vista architetturale.

L’unità di Controllo

- È l’organo che presiede a tutte le operazioni attraverso il pilotaggio dei componenti presenti nel processore.
- Ad esso spetta, ad esempio, l’interpretazione dell’istruzione che si trova di volta in volta nel registro IR; ad esso spetta abilitare alla lettura ed alla scrittura due registri tra i quali deve avvenire uno scambio di informazione.
- Tutte le fasi del ciclo del processore avvengono attraverso l’invio ai vari componenti di un insieme di impulsi di controllo, in una sequenza temporale ben precisa. Più precisamente, ad ogni colpo di clock le linee di controllo assumono un particolare stato; il susseguirsi dei diversi stati contribuisce all’esecuzione completa di un’istruzione.
- Per questo motivo si può dire che una singola istruzione in linguaggio macchina viene eseguita attraverso la opportuna composizione di più micro-operazioni.

La memoria

- Contiene un numero in genere molto elevato di registri nei quali vengono memorizzati i dati e le istruzioni di un programma.
- Il tempo impiegato per accedere ad un registro di memoria è generalmente molto superiore a quello impiegato per l'accesso ad uno dei registri del processore.
- È per questo motivo che, per quanto possibile, si tenta di utilizzare i registri interni per effettuare le operazioni, limitando gli accessi in memoria allo stretto necessario.
- Pur contenendo la memoria un numero molto elevato di registri, in ciascun istante temporale solo uno di questi è abilitato a partecipare ad operazioni di lettura o scrittura: quello il cui indirizzo è contenuto nel registro MAR.

Un esempio: esecuzione dell'istruzione MOVE (1)

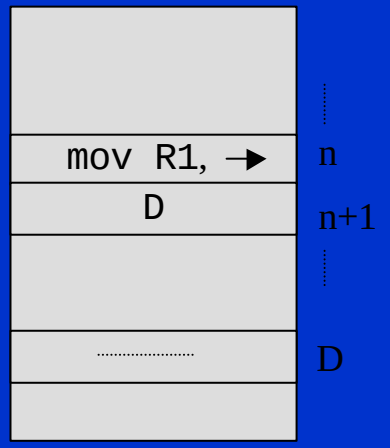
- Con riferimento al ciclo del processore, analizziamo come avviene lo scambio di informazioni, nell'ambito dell'architettura esposta, in un caso reale.

- Supponiamo che, durante la sua elaborazione, il processore si trovi ad un certo istante a dover eseguire un'istruzione che abbia lo scopo di spostare il contenuto corrente del registro generale R1 nella locazione di memoria avente indirizzo D. Rappresentiamo simbolicamente una tale operazione con il seguente comando:

```
mov R1,D
```

Un esempio: esecuzione dell'istruzione MOVE (2)

- Una tale istruzione, per essere eseguita, deve fare parte del bagaglio assembler del processore in questione e, come tale, essere opportunamente rappresentabile in linguaggio macchina. Si immagini, dunque, che l'istruzione sia rappresentata in memoria così come riportato in figura.

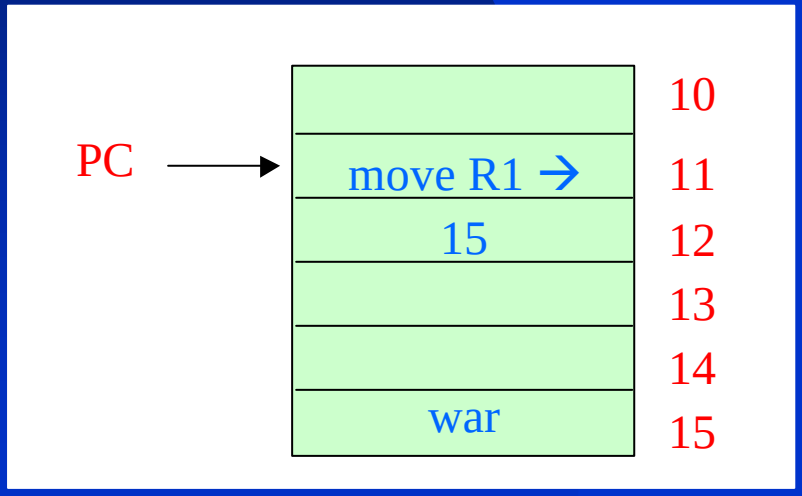


- Essa parte dalla n-esima locazione di memoria ma, non potendo essere contenuta completamente in una singola locazione, occupa anche la locazione successiva. In particolare, la locazione n-esima contiene una stringa di bit che verrà interpretata dall'unità di controllo come una

  "move la cui sorgente è il registro R1 e la cui destinazione è la locazione di memoria il cui indirizzo è contenuto immediatamente di seguito".

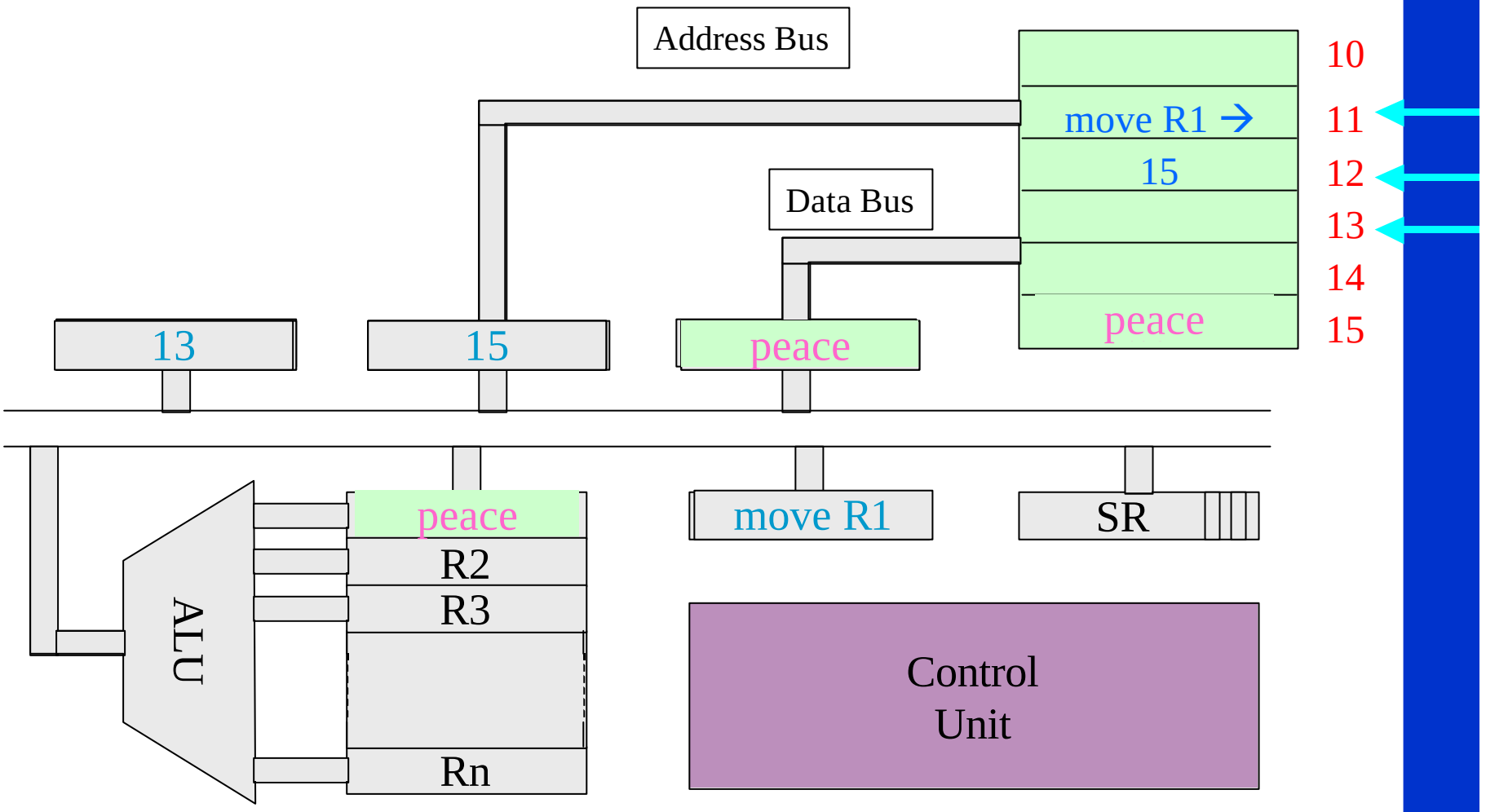
Un esempio: esecuzione dell'istruzione MOVE (3)

Supponiamo che l'istruzione sia contenuta in memoria nel modo seguente



1) PC → MAR
2) Inc(PC)
3) Memory Read (MDR=M[MAR])
4) MDR → IR
5) PC → MAR
6) Inc(PC)
7) Memory Read (MDR=M[MAR])
8) MDR → MAR
9) R1 → MDR
10) Memory Write (M[MAR]=MDR)

Esempio: esecuzione di un'istruzione



L'istruzione MOVE: fase *fetch*

- Trovandoci all'inizio della fase di fetch, bisogna prelevare l'istruzione dalla memoria: essa, per definizione, è contenuta all'indirizzo puntato dal registro PC.
- Il passo 1 copia il contenuto del PC nel MAR in modo da preparare la memoria ad un accesso alla giusta locazione.
- Il passo 2, in maniera del tutto generale, incrementa il PC in modo che esso punti alla locazione di memoria successiva, essendo questa la locazione da accedere nel seguito con la maggiore probabilità.
- Il passo 3 ordina alla memoria di scrivere sul Data Bus e, contemporaneamente, al MDR di leggere dal Data Bus. In seguito a questa operazione il MDR conterrà la prima parte dell'istruzione da eseguire.
- Essa, per essere interpretata, deve comunque essere trasferita nel IR. Questo avviene al passo 4. Una volta che l'istruzione è stata interpretata, l'unità di controllo "capisce" che l'istruzione è stata prelevata solo parzialmente e bisogna prelevarne un altro frammento.
- Qui la fase di fetch termina ed inizia la fase di preparazione degli operandi.

L'istruzione MOVE: fase *operand assembly*

- Nel passo 5, di nuovo il PC viene copiato nel MAR per permettere un nuovo accesso alla memoria.
- Immediatamente dopo, nel passo 6, viene nuovamente incrementato così da farlo puntare alla successiva istruzione, quella che verrà prelevata ed eseguita durante il ciclo successivo.
- Nel passo 7 la memoria viene letta e l'indirizzo D viene copiato nel MDR.
- Qui termina anche la fase di preparazione degli operandi ed inizia la fase di esecuzione vera e propria dell'istruzione.

L'istruzione MOVE: fase *execute*

- Dal momento che la scrittura deve avvenire proprio all'indirizzo ora contenuto nel MDR, questo viene copiato nel passo 8 sul MAR.
- Nel passo 9 il contenuto del registro R1 viene copiato nel MDR.
- Infine da qui, nel passo 10, viene copiato in memoria alla giusta locazione attraverso un'operazione di scrittura.
- Termina così anche la fase di esecuzione.
- A questo punto il PC punta alla successiva istruzione da eseguire, ed una nuova fase di fetch può avere inizio.